



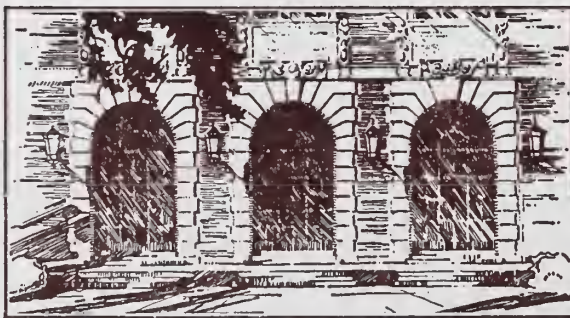
LIBRARY OF THE  
UNIVERSITY OF ILLINOIS  
AT URBANA-CHAMPAIGN

510.84

IL6r

no. 316-322

cop. 2



MATHEMATICS

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

To renew call Telephone Center, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

**BUILDING USE ONLY**

**JAN 18 1982**

JAN 18 1982



Digitized by the Internet Archive  
in 2013

<http://archive.org/details/interactivegraph318gear>

510.84  
Il6.2  
no. 318  
cop. 2

REPORT NO. 318

*Math*

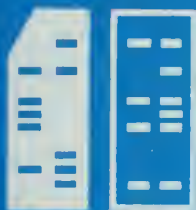
COO-1469-0115

AN INTERACTIVE GRAPHIC MODELING SYSTEM

by

C. W. Gear

April 1969



DEPARTMENT OF COMPUTER SCIENCE  
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS



Report No. 318

AN INTERACTIVE GRAPHIC MODELING SYSTEM\*

by

C. W. Gear

April 1969

Department of Computer Science  
University of Illinois  
Urbana, Illinois 61801

\* This report was supported in part by Grant U. S. AEC AT(11-1)1469.





## TABLE OF CONTENTS

	<u>Page</u>
1. INTRODUCTION. . . . .	1
2. GRAPHICAL SPECIFICATION . . . . .	8
2a. DEFINING BASIC ELEMENTS. . . . .	9
2b. SPECIFYING THE NETWORK . . . . .	13
2c. RESTRICTIONS ON THE ELEMENTS AND THE NETWORK. . . . .	14
2d. COMPOUND ELEMENT DEFINITION. . . . .	18
2e. DATA INPUT . . . . .	19
3. EFFICIENCY, NUMERICAL CONSIDERATIONS. . . . .	21
3a. STIFF DIFFERENTIAL EQUATIONS AND NONLINEAR ALGEBRAIC EQUATIONS. . . . .	23
3b. SPARSE MATRICES. . . . .	27
3c. GENERATING THE LINEAR EQUATIONS. . . . .	28
4. NETWORK ANALYSIS--SYMBOLIC MANIPULATION . . . . .	32
4a. SYMBOLIC SPARSE MATRIX INVERSION . . . . .	33
5. CONCLUSIONS . . . . .	46



## 1. INTRODUCTION

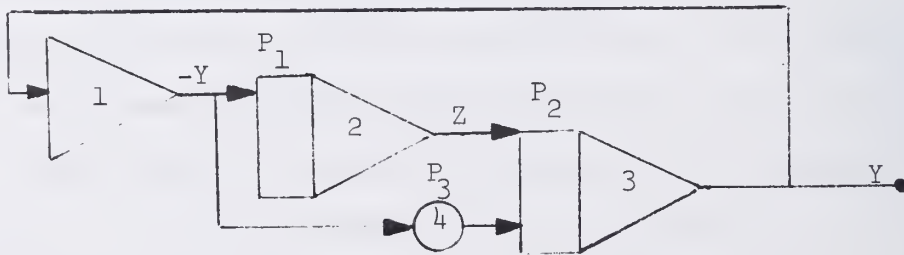
This report outlines the design of a general purpose graphic interactive modeling and simulation system in which the user can define the primitives of the system in the first stage, and then use them to model his problem in the second. This is similar to the compiler-compiler concept in that a language is used to define a further language. The first language allows the definition of the elements while the second consists of the use of these elements. It is unlike the compiler-compiler in the requirement that the first language must also be heavily user oriented, so that every user will be readily able to define his own elements during the development of his model, although he may make use of libraries of fairly common elements.

The use of graphics during all stages of system use is stressed as an important part of the interactive quality of the system, although it is certainly true that with suitable input and output conventions, the system could be operated from a typewriter terminal or through batch.

In any modeling system the user is provided with a number of modeling blocks (elements) which follow certain laws (that is, obey certain equations) and he may connect these in various ways. These connections may be of a topological nature, that is, elements may be joined together at certain connection points (called terminals below) or may satisfy some global relation (for example, have a common ambient temperature). Existing modeling systems provide predefined elements, although they may allow the user to define additional elements of a similar kind. Examples of such systems exist in simulation (e.g. the IBM CSMP system<sup>[2]</sup>) in which the elements represent typical analog computer elements, and in network analysis (e.g. the Electronic Circuit Analysis Program ECAP). These systems are tailored for particular classes of problems, and cannot be used on larger classes without a great deal

of analysis on the part of the user. The purpose of the proposed system is to make it possible for the user to define any type of element and the way in which it can interact with other elements. It should not be restricted to single variable elements as are simulator elements, or to two terminal elements as are electrical network elements. Thus, the elements might represent atoms reacting dynamically with a number of near neighbors in a lattice, cars passing through an intersection, or complex electrical network elements with many terminals.

A typical simulation program will allow the user to join a number of elements together in a network such as



In this diagram, element 1 is an inverter, elements 2 and 3 are integrators, while element 4 is a gain. These elements are characterized by the fact that the input and output "wires" each "carry" the value of a single real number, and that they are directional elements; that is, they have distinct inputs and outputs. Thus, the input to element 1 carries the value  $y$ , so its output carries the value  $-y$ . Since element 2 is an integrator, its output  $z$  is given by the expression

$$z = P_1 + \int_0^t (-y) dt$$

The  $P_1$  is a parameter that can be set as an initial value. The output of element 4 is  $P_3$  times its input, or  $-P_3 y$ . The output of element 3 is the integral of the sum of its inputs with the initial value of  $P_2$  so we have

$$y = P_2 + \int_0^t (z - P_3 y) dt$$

On differentiating twice we get

$$\frac{d^2 y}{dt^2} = -y - P_3 \frac{dy}{dt}$$

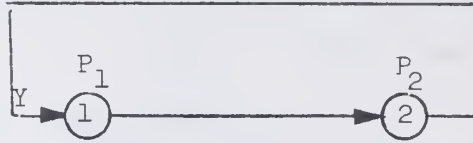
$$y(0) = P_2$$

$$\frac{dy}{dt}(0) = P_1 - P_2 P_3$$

Thus, the particular network can be used for integrating a linear homogeneous second order differential equation. In addition to these elements, many others which combine the inputs in both continuous and discontinuous fashions are available and most systems will allow the user to define some new elements by means of available procedure oriented language subroutines. In all cases, however, the elements are uni-directional; that is, they have inputs and a single output each of which carries only a single value. The output of any element may be connected to any number of input elements, but not to another output element, since then the number carried on the "wire" would be doubly defined.

These requirements also lead to other types of restrictions, the most important being that no loop may exist that does not contain an element that causes a delay. A loop is any closed path such as occurs in Figure 1 going from element 1 to 2 to 3 and back to 1. A delay can be caused by a

delay element that states that the output is the value of the input at a given earlier time or an integrator. This type of restriction insures that the situation shown below cannot occur.



This network defines  $y$  to be  $P_1 P_2 y$ , which means that  $y$  is zero unless  $P_1 P_2 = 1$ . Most digital simulators retain this restriction because it simplifies the method of solution--on an analog computer such a connection is not practical because of inherent delays in the physical realization of the element.

Systems have been produced to interface graphic terminals with simulator programs. See, for example, Baskin<sup>[1]</sup>. The user is presented with a set of predefined elements, and is allowed to define additional elements that are of the same general class.

Network analysis programs use two terminal elements that are bi-directional; that is, there is no distinction between input and output. Whereas the nodes of a network for a simulation system were governed by the fairly simple requirement that a single output signal must be connected, and that this determines the value of all connected inputs, the nodes of the electrical network are governed by Kirchoff's laws--namely, that the voltage of all lines connected to the node is the same and that the sum of the currents entering the node from all connected elements is zero. The former can be replaced by the statement that the sum of the voltage drops across elements

in any closed loop is zero. These facts are used to take the equations that govern each element and derive a system of differential equations. A knowledge of the element equations is built into the network analysis systems and cannot be changed.

In a similar manner, structural networks consisting of members (columns and beams, etc.) are connected together at nodes. The individual members are governed by equations (Hooke's law relating the elastic "stretch" to the applied force, strain and stress to give them their correct terms, for example) while the ends of the beam are characterized by twelve physical quantities, the six components of deflection and the six components of force. The nodes imply the equation that the sum of all of the forces acting on a node is zero, component by component, and that the deflection of all members connected to a node is the same. Thus, there is a direct analogy between current and force and between voltage and deflection. In fact, the same methods of analysis can be used for the two problems.

In the analog computer, a restriction that no loops could be specified that did not include a delay was enforced. No such restriction is reasonable in the case of electrical or structural networks; consequently, the network analysis programs contain techniques to analyze such loops and to calculate their overall effect. The network analysis programs can perform this extra service because they "know" the type of behavior possible in each element, and can, therefore, apply known techniques for solving the resulting equations. The simulation program users are allowed to specify arbitrary elements by means of computer programs so it is difficult to determine when a closed loop will have a solution.



The proposed system will allow arbitrary elements to be specified and used in networks, so it will contain standard simulators and network analyzers as a subset. Naturally, the generality will lead to a loss of speed and it is appropriate to ask at this point how this will be overcome and how the system will be able to handle the more complex networks that are not allowed in existing systems. Because the system is not yet fully implemented, this can only be partially answered. New numerical techniques for the solution of ordinary differential equations<sup>[5]</sup> will be used. These have produced speed increases of several orders of magnitude for one class of problems encountered in simulation. The use of sparse matrix techniques and an initial symbolic processing of the equations can replace repeated numerical passes over parts of the equations by a single symbolic pass. This is discussed in some detail in Section 4. The matrix techniques will also be needed for the detection of incorrectly specified networks. Because the usual restrictions of simulation and network analysis systems are dropped, it is possible for the user to give networks with many solutions or none. It is possible to detect this in almost all cases with techniques given in Section 4. The unsolved problem is how to guarantee that all cases are detected, and more important, how to relate detected under or over specifications to the original user input. When a user of an analog computer simulator fails to connect an output to one of the nodes, the simulator can tell him of the error in those words. When the system is represented by a large set of differential equations, the system will only detect that there are more variables



than equations. The solution will require some restrictions on element specification, but it is not clear what they are at this time.

Section 2 will discuss the graphical interface between the user and the system, Section 3 the numerical algorithms used and Section 5 indicates what the current state of the art is by describing the capabilities of the system that can (and are) being implemented. without further theoretical advances.

## 2. GRAPHICAL SPECIFICATION

An interactive graphic display will be used to specify the elements used and the network connections. The use of graphics during the basic element definition stage is for mnemonic value only, whereas the graphical description of the element connections provides a more concise description of the network than most other techniques. Graphics will also have an obvious application during the data input and output stages.

In very complex networks in which some sections behave almost independently of the others, it may be helpful to allow the user to point out subareas that can be separately analyzed.

## 2a. DEFINING BASIC ELEMENTS

An element will be represented by a mnemonic picture which is drawn by the user. The element contains a number of parts:

1. An arbitrary line drawing.

This is the mnemonic for the element which is used for identification purposes by the user. It could consist of the drawing



for example. There is no significance to any of the lines, so the picture is retained within the graphic section entirely.

2. Arbitrary character strings.

These are part of the line drawing and have only mnemonic value. The figure above might also contain the string "1 WATT", presumably to identify it as a one watt component. This naming should not be confused with parameter specifications described below.

3. Labeled terminal points.

Any of the points in the line drawing may be labeled. We will use labels 1, 2,... etc. Each label must be unique. The drawing above could have its left and right ends labeled 1 and 2, respectively. These terminals will be the connection points of the element.

#### 4. Descriptive equations.

These specify the action of the element. They can involve variables defined on the terminals and variables internal to the element. If the figure above refers to a resistor which depends on temperature, the equations might be

$$E(2) = E(1) + (R_0 + R_1 * T) * I(2)$$

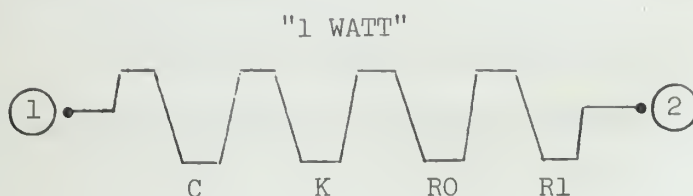
$$I(1) = -I(2)$$

$$T' = (E(2) - E(1)) * I(2) * K - C * (T - T_0)$$

Here the resistance is assumed to be a linear function of the element temperature  $T$ , which itself is determined by a first order differential equation involving the voltage drop  $(E(2) - E(1))$  and the current  $I(2)$ .  $T_0$  is the ambient temperature, while  $C$ ,  $K$ ,  $R_0$ , and  $R_1$  are constants for the device. Prime denotes differentiation.

#### 5. Parameters.

In most applications, elements such as the one specified will be used repeatedly with different values of the device constants. Therefore, these must be parameters and be given values each time they are used. Consequently, at the specification stage the user can also place labels on the figure corresponding to the symbols used in the equations. At the time of use, these will be replaced by numeric values or expressions involving other variables. Thus, the complete specification for this example is



$$E(2) = E(1) + (RO + RL * T) * I(2)$$

$$I(1) = -I(2)$$

$$T' = (E(2) - E(1)) * I(2) * K - C * (T - T_0)$$

Variables were assumed to be associated with each of the nodes. In the example above, voltage  $E$  and current  $I$  was assumed on each of the two nodes. In a general problem, there can be one or many variables associated with each node. A loose jointed three dimensional structure has six, for example, while an analog computer has one. Thus, the system must be told how many variables appear on each terminal and what their names are. In a later step, terminals of several elements will be joined together in a network node. Rules are needed for generating relations between the variables on the connected terminals. In all problems studied by this writer, only two types of rules have been necessary: Kirchoff's current and voltage laws. Thus, we will identify variables as being  $I$  type or  $E$  type depending on which rule they follow.

It may also be possible for different types of terminals that may not be interconnected to occur. (This could arise because of different signal levels, for example.) In that case the user would like the option of specifying which types of terminals are to be used. Therefore, before any elements are specified, the user will have to answer the following questions:

HOW MANY NODE TYPES WILL YOU USE?

NAME ALL  $E$  TYPE VARIABLES ON NODE TYPE 1.

NAME ALL  $I$  TYPE VARIABLES ON NODE TYPE 1.

(The latter two questions are repeated for each node type.)

In the example above the answers would be I, E, and I.

The system will then proceed through the sequence of questions:

DO YOU WISH TO SPECIFY A NEW ELEMENT?

If the answer is yes, the opportunity to make an arbitrary line drawing is provided. (It is also possible to include character strings.) An escape from this will lead to the question:

DO YOU WISH TO INDICATE A TERMINAL POINT?

A yes answer will yield the opportunity to indicate a point in the figure. It is labeled by the next available integer, starting at 1, and the user is asked

WHAT IS ITS NODE TYPE?

(unless only one node type was specified).

When all terminal points have been specified, the user is asked:

DO YOU WISH TO SPECIFY PARAMETERS?

If yes, he indicates their positions by light-pen and inputs their names. Finally he is asked to

PLEASE SPECIFY THE ELEMENT EQUATIONS

He will type them in the manner indicated in the above example.

## 2b. SPECIFYING THE NETWORK

When all desired elements have been specified, the user will be in a position to use them in a network. Each element will appear as a light-button (menu item) on the display. Instances of the elements may be selected and positioned on the display. Terminals of elements may then be connected to form nodes of the network. If more than one node type is specified, the system will restrict interconnections to terminals of the same type. As instances are connected, the system will be producing systems of equations which describe the network.

As an instance of an element is selected, the user is asked to specify the values of each parameter. This can be by numeric input (assigning the value of a resistor, for example), or the parameter could be given another symbolic value for later specification. (An element could depend on its temperature which is later defined to be the temperature of another element, for example.)



## 2c. RESTRICTIONS ON THE ELEMENTS AND THE NETWORK

The result of a set of element descriptions and a network which uses them will be a set of algebraic and differential equations. It is evident that it must be possible to solve these if the network is to have any meaning. (There are cases in which the solution is not unique but for which such non-uniqueness is unimportant. These will be discussed in Section 3.)

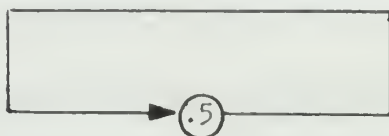
Consequently, it must be possible to determine whether or not the system can be solved. This is probably the most severe problem introduced by the generality of the processor. There is naturally a trade-off between the difficulty of determining solvability and the freedom allowed the user. In existing systems, the number of variables allowed on terminals, the type of relations which can be given and connection restrictions may be built in. A simulator system, for example, allows one variable of E type on each terminal. It may have many terminals and some terminals may be distinguished as output terminals. One and only one output terminal must be connected to each node of the network. Furthermore, many simulation systems restrict the network connections to avoid "unnatural" feedback connections--those loops that contain no delay elements. Thus if the circle represents a gain element whose output is  $g$  times the input defined by



$$E(2) = g * E(1)$$

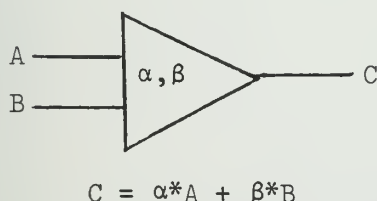


the network

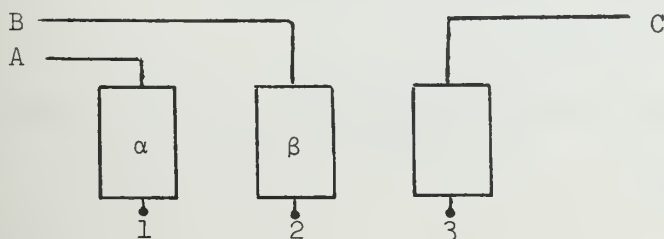


is not one that would normally be allowed on an analog computer, although algebraically it serves to define  $E(1) = E(2) = 0$ . (If, however, the gain were unity, it would not define the value of  $E$  so in an algebraic sense it would retain its initial value. It would be unwise to use such a subnetwork in an analog computer program!) On the otherhand, electrical networks restrict the user to two terminal elements (usually) with two variables on each terminal. It is also known that the  $I$  type variable sums to zero for the element and that the  $E$  type variable enters only via the difference between the values on the two terminals. There are no distinguished output terminals, although there are restrictions against connecting voltage generators in parallel or current generators in series.

(Parenthetically, it should be noted that simulator elements can be viewed as a set of two terminal elements. Thus, the element



can be viewed as three two-terminal elements



where the terminals 1, 2, and 3 are "grounded" in use, and the equations are

$$C - E(3) = \alpha*(A - E(1)) + \beta*(B - E(2))$$

The I variable for each terminal can be ignored.)

If no restrictions are placed on the definition of elements, it is necessary to wait until the complete network has been specified, collect all relevant equations, and then attempt to determine whether or not they are singular. By placing some restrictions on the definition of new elements, the process of determining non-singularity can be reduced. The system requires that the restrictions given below are adhered to in the definition of elements and their connection in networks. Alone they are not sufficient to ensure that the equations are non-singular, but they ease the analysis of the network.

### Restrictions on Elements

The equations for the internal variables  $u$  may take the forms

$$(i) \quad u' = f(\text{all variables})$$

or

$$(ii) \quad u = a(\text{all other variables})$$

where the prime denotes differentiation with respect to time. There must be exactly one such equation for each internal variable. The terminal variables  $v$  must satisfy equations of the form

$$v = a(\text{all other variables})$$

where each  $v$  may only appear on the lefthand side of one such equation (but may not appear on any).

An option in which the equality sign is replaced by the left arrow ( $\leftarrow$ ) assignment operation allows the user to indicate an element which "forces" the value of a terminal variable. Only one such terminal may be connected to any node. Thus, the equality sign is used in the strict sense rather than the procedural language assignment sense.

#### Network Restrictions

Restrictions which prevent multiple assignment of variables on a node by the left arrow are obvious. Similarly, equations of the form

$$E(1) = 0$$

are equivalent to assignments, but can easily be detected and so treated. More difficult are equations of the form

$$E(1) = E(2) + 1$$

(that is, voltage generators). Two of these may not be connected in parallel, but recognition of this and more complicated forms have to be left to the analysis stage described in Section 4.

## 2d. COMPOUND ELEMENT DEFINITION

It is desirable to be able to use previously defined elements in the definition of new elements. The steps in this process are as follows:

1. Construct a network from existing elements.
2. Identify nodes in this network as terminals of the compound element.
3. Specify any additional equations connected with the element. These may include new element variables which could be parameters in some of the constituent elements. For example,  $T_0$ , the ambient temperature in the resistor example above, may vary for the component element due to heating effects of all of the elements. It could also be specified by differential equations.
4. Replace the drawing of the network by a new line drawing that serves to identify the element, and indicate the terminal points on the line drawing.
5. Specify parameters of the compound element, and indicate where they are to be placed.

The new drawing will then be available as an element for the construction of further networks.

## 2e. DATA INPUT

Prior to the simulation of the network, values must be provided for any variables as yet unspecified and certain nodes may have to be set to given input values. Nodes can always be set by defining "voltage source" elements with a given time dependent behavior, so the problem is equivalent to one of specifying arbitrary function values for some variables. Thus, we may have the element



$$E(1) = E(2) + F$$

where  $F$  is a variable and we wish to assign  $F$  a value. If it is a simple function, it can be expressed algebraically, for example

$$"F = 3*\text{SIN}(T)"$$

However, we wish to allow more general inputs. It must be possible to specify an arbitrary function  $F(Y)$  by means of a table or a graph, and to use this function in the definition of a variable. Consequently, we allow the input of tables or single valued graphical functions and use the information to determine an approximation to the function. It is not obvious what approximation should be used, but a Spline function would have the advantage of derivative continuity. It may also be desirable to allow discontinuous functions although these slow the simulation process down because of numerical problems with high order methods.

When all parameters have been specified, it is necessary to determine the initial conditions. The user may wish to specify some, others can then be determined by performing a simulation with constant inputs until

the system settles (if it is stable). An integration with time as the independent variable can then be used to follow the system behavior. Values of terminal or element variables or expressions containing them can be specified as output which can be obtained in tabular or graphic form.

It is obviously desirable to be able to repeat analysis for different parameter and initial values, so the final step is to allow statements in a conventional programming language to be used to write loops and to call the simulation system.

### 3. EFFICIENCY, NUMERICAL CONSIDERATIONS

When a complete network has been specified, the equations representing each of the elements can be collected. These equations will involve the variables on the elements and on their terminals. The equations consist of a set of linear and nonlinear algebraic and differential equations. They will be written in the form

$$u'_i = f_i(u_j, v_j, w_j, t) \quad (3.1)$$

$$v_i = g_i(u_j, v_j, w_j, t) \quad (3.2)$$

$$R_{ij}w_j = (P_{ij}u_j + Q_{ij}v_j) \quad (3.3)$$

where the prime represents differentiation with respect to time, the  $u_i$  are the set of variables defined on the elements which satisfies differential equations, the  $v_i$  are the set of variables that appear either on elements or on terminals that are specified by nonlinear equations and the  $w_i$  are the remaining variables. Summation is presumed to occur on the  $j$  subscript in the third set of equations.

If the network is improperly specified, then these linear equations may have no solution or may possess a nonunique solution. There may be more or fewer equations than there are unknowns. (In other words,  $R$  may or may not be square and even if square may not have an inverse.) However, even if the equations are properly specified, there may not be a unique solution. If, for example, the variables on the nodes are actually current and voltage and all elements only specify the voltage drop across the element (there is no element that specifies the actual voltage of one terminal), then the value of the voltage of at least one



node will be undetermined. All other nodes may be given as a level relative to that. Similarly, if all current sums for elements are zero, at least one of the Kirchoff current laws for the nodes is redundant. Thus, although the matrix  $R$  may be square, it could be singular and the equations have many solutions. For this section we will assume that  $R$  is square and nonsingular. The difficulties will be discussed in Section 4.



### 3a. STIFF DIFFERENTIAL EQUATIONS AND NONLINEAR ALGEBRAIC EQUATIONS

The resulting system of differential equations is frequently stiff. This means that there are widely separated time constants present. This causes most common methods to be extremely slow, using steps which are of the order of the shortest time constant present even though the components due to these time constants have been damped out for a long time. Recently some new methods have been used very successfully on such problems.<sup>[5]</sup> They are a class of methods of varying orders of the multistep type, but can be organized so that both the step and order can be chosen automatically in order to make the step as large as possible while controlling the single step error. They require no special starting procedure because it is part of the step and order control mechanism. These methods will be used in the system.

If a system of the form (3.1) is given--we will ignore the other variables for the moment--it is necessary to calculate an approximation to the Jacobian  $J = \partial f_i / \partial u_j$  in order to solve the corrector formula. The iteration is given by

$$u_{ik}^{(m+1)} = u_{ik}^{(m)} + b_k \sum_j [I + hb_1 J]_{ij}^{-1} [u_j^{(m)} - hf_j(u_{il}^{(m)}, t)]$$

where  $u_{ik}$  is the calculated approximation to

$$\frac{h^{k-1} d^{k-1} u_i}{(k-1)! dt^{k-1}}$$

and  $h$  is the time step. These approximations are used to estimate (predict) the value of  $u_{ik}$  at the next time step. The coefficients  $b_k$

determine the method. The corrector iteration actually amounts to a Newton-Raphson iteration if the Jacobian  $J$  is re-evaluated at each iteration, otherwise it is a chord method. As long as it converges, it converges to a solution of the corrector, so it is not necessary to have a precise value of  $J$ . The system of nonlinear algebraic equations also has to be solved at the same time. These can also be handled by a Newton or a chord method in a similar way. Suppose that a good first approximation to  $v_{ik}$  (defined in the same way as  $u_{ik}$ ) is known. Then the iteration

$$v_{ik}^{(m+1)} = v_{ik}^{(m)} + c_k \sum_j \left[ I + c_l \left\{ \frac{\partial g_i}{\partial v_j} \right\}_{ij}^{-1} \right] [v_{il}^{(m)} - g_i(v_{jl}^{(m)}, t)]$$

converges to the set of numbers  $v_{ik}$  in which  $v_{il}$  will be the solution of the algebraic equations, while the remaining  $v_{ik}$  will be approximations to the scaled derivatives. (They will be in error by the first neglected derivative after sufficient steps.) Again the Jacobian need not be re-evaluated at every step of the iteration, because it only affects the rate of convergence, not the final solution. The reason for calculating the additional derivatives is that very accurate first approximations to the value at the next time step can be obtained so by prediction that very few iterations are needed. (One or two is typical.) The constants  $c_k$  determine the order of the method, that is, how many of the scaled derivatives are needed. A combination method in which the corrector equation and the algebraic equations are solved at the same time is given by calculating the matrix

$$Z = \begin{bmatrix} I + b_1 h \underline{f}/\partial \underline{u} & b + h \partial \underline{f}/\partial \underline{v} \\ c_1 \partial \underline{g}/\partial \underline{u} & I + c_1 \partial \underline{g}/\partial \underline{v} \end{bmatrix}^{-1}$$

and forming the correction factors

$$\underline{d} = Z \begin{bmatrix} \underline{u}_2^{(m)} - h \underline{f}(\underline{u}^{(m)}, \underline{v}^{(m)}, t) \\ \underline{v}_1^{(m)} - \underline{g}(\underline{u}^{(m)}, \underline{v}^{(m)}, t) \end{bmatrix}$$

where  $\underline{u}_2$  is the vector formed from  $u_{i2}$ , while  $\underline{v}_1 = \underline{v}$  is the vector formed from  $v_{i1}$ . Products of the elements of  $\underline{d}$  with the vectors  $b_k$  and  $c_k$  can then be added to the matrices  $u_{ik}^{(m)}$  and  $v_{ik}^{(m)}$  to form the  $(m+1)$ -st iterate.

It was assumed above that the vectors  $\underline{f}$  and  $\underline{g}$  are functions of  $\underline{u}$  and  $\underline{v}$  only. They are also functions of  $\underline{w}$  and this must be taken into account in calculating  $Z$ . Since we assumed that  $R$  is nonsingular we can write

$$\underline{w} = R^{-1}(P\underline{u} + Q\underline{v}) \quad (3.4)$$

We will write

$$\frac{\partial \underline{f}}{\partial \underline{u}} = A, \quad \frac{\partial \underline{f}}{\partial \underline{v}} = B, \quad \frac{\partial \underline{f}}{\partial \underline{w}} = C$$

$$\frac{\partial \underline{g}}{\partial \underline{u}} = D, \quad \frac{\partial \underline{g}}{\partial \underline{v}} = E, \quad \frac{\partial \underline{g}}{\partial \underline{w}} = F$$

We may then write

$$Z = \begin{bmatrix} I + b_1 h(A + CR^{-1}P) & b_1 h(B + CR^{-1}Q) \\ c_1(D + FR^{-1}P) & I + c_1(E + FR^{-1}Q) \end{bmatrix}^{-1}$$

We also need to calculate the functions  $\underline{f}$  and  $\underline{g}$  for a given  $\underline{u}$ ,  $\underline{v}$ , and  $t$ . To do this we must evaluate (3.4). The linearity of  $\underline{u}$  and  $\underline{v}$  in this is not important, so we could have allowed (3.3) to be written

$$R\underline{w} = \underline{q}(\underline{u}, \underline{v}) \tag{3.5}$$

where

$$P = \frac{\partial \underline{q}}{\partial \underline{u}} \text{ and } Q = \frac{\partial \underline{q}}{\partial \underline{v}}$$

### 3b. SPARSE MATRICES

Typical networks contain many elements and even more equations involving only a few variables. Consequently, the work involved in calculating  $Z$  and evaluating (3.4) may be the limiting factor in speed. For this reason we use sparse matrix techniques. Since the multiplication of  $Z$  by a vector occurs in every iteration step, the objective is to compile code to perform this as rapidly as possible, and never actually generate the elements of  $Z$  explicitly. The value of the Jacobian may only be recalculated periodically (when the step size changes, for example) so that the calculation that is dependent only on the Jacobian should be done separately and not repeated. This is the subject of the next section.

## 3c. GENERATING THE LINEAR EQUATIONS

The linear equations can be solved directly by matrix inversion (using a sparse matrix technique) so it is desirable to place as many equations in the form (3.3) as possible. Since we allowed them to take the more general form (3.5) we must give an algorithm for determining which set of variables is  $\underline{w}$ . The set  $\underline{u}$  can be determined immediately by examining the element equations. The remaining element equations and node equations must first be scanned to remove the trivial relations, and then scanned to generate the nonlinear equations. When a nonlinear equation of the form

$$x = x(\text{all variables})$$

is given we must check to see if all the variables that appear nonlinearly in the equation have already appeared on the lefthand side of the differential or nonlinear equations already generated. If not, one of these variables can be used, say  $v_j$ , and the equation written as

$$v_j = v_j + x(\text{all variables}) - x \tag{3.6}$$

(In fact, it is not necessary to add the  $v_j$  to both sides, but is is helpful for expository purposes to make it similar in appearance to the differential equations.) If all of the variables appearing nonlinearly have already appeared on the lefthand side, the equation can be saved as a candidate for a "linear" equation. When the scan has been completed, a set of such candidates remains. (Parenthetically, we note that the choice of which variable  $v_j$  to place on the left of (3.6) may affect the number of equations left as possible linear

equations. Thus, if we have the system

$$0 = x_1 + x_2 * x_3 + 1$$

$$0 = x_1 + x_2 ** 2 + x_3$$

$$0 = x_1 + 1/x_2 - x_3$$

and we choose to write the first as

$$x_2 = x_2 + x_1 + x_2 * x_3 + 1$$

the remaining two can be written in the form

$$x_1 + x_3 = x_2 ** 2$$

$$x_1 - x_3 = 1/x_2$$

whereas had we started with

$$x_3 = x_3 + x_1 + x_2 * x_3 - 1$$

we would also have called the second nonlinear and written

$$x_2 = x_2 + x_1 + x_2 ** 2 + x_3$$

leaving one linear equation. The solution technique will give the same answers, but the speed may be different. One of the questions yet to be investigated is that of optimizing the assignment.)

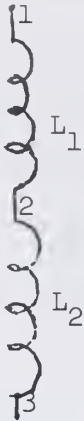
The remaining equations can be written in the form

$$R\underline{w} = q(\underline{u}, \underline{v})$$

but R may be singular. It is therefore necessary to remove some of the



singularities. Consider the part of a network below



representing two inductors in a series. These will eventually reduce to the equations

$$I(1)' = (E(1) - E(2))/L_1$$

$$I(2)' = (E(2) - E(3))/L_2$$

and

$$I(1) - I(2) = 0$$

These serve to determine  $(E(1) - E(3))$  and to determine  $E(2)$  in terms of  $E(1)$  and  $E(3)$  (independent of  $I(1)$  and  $I(2)$ ). Unfortunately the scheme given above would place  $I(1)$  and  $I(2)$  in the set  $\mathbf{u}$  and  $E(2)$  in the set  $\mathbf{w}$ . The third equation would be called linear so its row of  $R$  would be zero, making  $R$  singular. If the third equation is placed in the "nonlinear" class and  $E(2)$  placed in the class  $\mathbf{v}$  by writing

$$E(2) = E(2) + I(1) - I(2)$$

the numerical technique will work for nonzero step sizes  $h$ .

Consequently, rows and columns of  $R$  must be removed. This will have to be detected during the symbolic inversion process discussed in the next section.



An alternative technique that can be used in this case is to recognize the problem during the scan to remove trivial equations. This would try and replace  $I(2)$  by  $I(1)$  and would lead to two differential equations for  $I(1)$ . The second could be converted to the algebraic equation

$$(E(1) - E(2))/L_1 = (E(2) - E(3))/L_2$$

which will give one differential equation and one linear equation for this section.

This latter technique cannot eliminate all of the singularities in  $R$ . For example, an element between  $L_1$  and  $L_2$  above could have given a nonlinear relation between  $I(1)$  and  $I(2)$ . Thus, we must be prepared to detect singularities in  $R$  during the analysis.

#### 4. NETWORK ANALYSIS--SYMBOLIC MANIPULATION

A number of steps have to be taken symbolically because a key to the system is the ability to perform as much of the matrix inversion prior to the integration as possible and to be able to detect singularities in the system. Once numbers have been inserted we cannot guarantee that zeros arise from singularities rather than cancellation.

The first step is to develop the Jacobian matrix for all equations. This requires a partial differentiation as performed in ODESSY<sup>[4]</sup>. It is then generally simple to detect constants which indicate linear terms. (It may not be possible to find all such constants because full automatic simplification is not possible.)

This information is used to generate the tentative set of linear equations, and hence the matrix  $R$ . The next step is to try and invert  $R$  "symbolically." If singularities are found, they must be removed by taking out the appropriate rows and columns and placing them in the set of nonlinear equations. There is an exception to this: If during the reduction of  $R$  a zero is encountered, and the righthand side arising from  $p(\underline{u}, \underline{v})$  is also identically zero, it means that one of the variables in  $w$  can be arbitrarily specified and that the zero equation can be ignored completely. This occurs, for example, when all elements involve only a voltage difference and all current sums are zero. One or more of the Kirchhoff current equations can be eliminated, and one or more of the voltages can be chosen arbitrarily. In this case, the equation need not be placed in the nonlinear group.

## 4a. SYMBOLIC SPARSE MATRIX INVERSION

We are given a matrix  $R$  whose elements may be zero, small integers, constants, or symbolic expressions. The problem is to solve the system

$$RX = B$$

where  $R$  is  $K \times N$ ,  $X$  and  $B$  is  $N \times M$ ,  $M \geq 1$ . Typically, the number of nonzero elements of  $R$  is small ( $2N$  to  $5N$ ),  $N$  is large ( $> 100$ ), and  $M$  small (often 1). The elements of  $B$  may also take one of the four forms cited. The values of the "variables" appearing in the symbolic expressions will be specified, at which time the value of  $X$  is needed. The goal is to do as much "preprocessing" as possible. Two situations need to be considered in the analysis. In the first the values of  $B$  will change very frequently, while the values of  $R$  will only change infrequently. In the second, we need to calculate a third  $M \times M$  matrix of the form

$$D = CR^{-1}B$$

where  $C$  is  $M \times N$ , and in this case,  $R$  may be singular (so that  $R^{-1}B$  can be interpreted as any solution of  $RX = B$ ,  $B$  being assumed to have the appropriate properties for such a solution to exist). If rows of  $R$  lead to singularities such that no solution exists, these rows must be identified and removed from  $R$ . Since  $D$  itself will later take part in a similar problem, the solution for  $D$  should be as non-numeric as possible--by this is meant that all elements that will be of the form of small integers or simple symbolic expressions should be carried that way. This will be clearer after reading the example following.

EXAMPLE

Suppose we have

$$\begin{array}{rcl}
 R(I,1) & = & 1 \quad -1 \quad 0 \quad 0 \quad S+T \\
 R(I,2) & = & 0 \quad L \quad 0 \quad 0 \quad -L \\
 \text{---} & & R \quad -R \quad 1 \quad 0 \quad 0 \\
 & & 1 \quad 0 \quad 0 \quad 1 \quad 0 \\
 R(I,5) & = & 0 \quad 1 \quad 0 \quad 0 \quad 1
 \end{array}$$

and

$$\begin{array}{rcl}
 B(1) & = & 1 \\
 B(2) & = & 0 \\
 \text{---} & & C \\
 & & 1 \\
 B(5) & = & D
 \end{array}$$

We will identify  $R(6,J)$  with  $B(J)$  so that  $R(I,J)$  is a  $5 \times 6$  matrix.

The Gauss elimination back substitution method (without concern for best pivots) leads to the following steps to convert the elements of  $B$  to the elements of the solution  $X$ :

$$-R \cdot R(I,1) + R(I,3) \rightarrow R(I,3) \quad I = 1, \dots, 6$$

Note that the symbolic operation for  $I = 2$  should give

$$\begin{aligned}
 & (-R) \cdot (-1) + (-R) \\
 & = R \cdot 1 - R \\
 & = R - R \\
 & = 0
 \end{aligned}$$

$$-R(I,1) + R(I,4) \rightarrow R(I,4)$$

Note that this makes  $B(4) = -1 + 1 = 0$  since we can do exact arithmetic on small integers.

These two steps reduce the  $5 \times 6$  R to

$$\begin{array}{cccccc} 1 & -1 & 0 & 0 & S+T & 1 \\ 0 & L & 0 & 0 & -L & 0 \\ 0 & 0 & 1 & 0 & -R*(S+T) & C-R \\ 0 & 1 & 0 & 1 & -(S+T) & 0 \\ 0 & 1 & 0 & 0 & 1 & D \end{array}$$

Divide  $R(I,2)$  by  $L$

Note that this makes  $R(5,2)$  into  $-1$  if symbolic arithmetic is performed. Note that  $B(2)$  is zero, so is not changed.

$$-R(I,2) + R(I,4) \rightarrow R(I,4)$$

Note that  $B(2)$  is zero, so  $B(4)$  is not changed.

$$-R(I,2) + R(I,5) \rightarrow R(I,5)$$

Note that  $B(5)$  is not changed since  $B(2)$  is zero, and that  $R(5,5)$  becomes  $-(-1) + 1 = 2$ , exactly.

Divide  $R(I,5)$  by 2

These steps reduce R to

$$\begin{array}{cccccc} 1 & -1 & 0 & 0 & S+T & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -R*(S+T) & C-R \\ 0 & 0 & 0 & 1 & -(S+T)+1 & 0 \\ 0 & 0 & 0 & 0 & 1 & D/2 \end{array}$$

The back substitution process is as follows

$$\begin{aligned}
 (S+T-1)*R(I,5) + R(I,4) &\rightarrow R(I,4) \\
 R*(S+T)*R(I,5) + R(I,3) &\rightarrow R(I,3) \\
 R(I,5) + R(I,2) &\rightarrow R(I,2) \\
 -(S+T)*R(I,5) + R(I,1) &\rightarrow R(I,1) \\
 R(I,2) + R(I,1) &\rightarrow R(I,1)
 \end{aligned}$$

which reduces R to

$$\begin{array}{ccccc}
 1 & 0 & 0 & 0 & 0 & 1-(S+T)*D/2 + D/2 \\
 0 & 1 & 0 & 0 & 0 & D/2 \\
 0 & 0 & 1 & 0 & 0 & C-R + R*(S+T)*D/2 \\
 0 & 0 & 0 & 1 & 0 & (S+T-1)*D/2 \\
 0 & 0 & 0 & 0 & 1 & D/2
 \end{array}$$

The solution X appears in the last column.

If this were to be done for large systems, it is evident that the symbolic forms would get unreasonably complex, even if optimum simplification were to be performed. Since the objective is to get fast code which means that common expressions should be factored out, it is sensible to remove them at an early stage in the process. Because, in this example,  $R(1,5) = S+T$  is already complex, it might be better (it is in this case) to rename it  $R_{15}$  which is equivalent to saying that, "We have a way of calculating this element but we do not know anything about its relation to other elements in the matrix." This may mean that if  $S+T$  appeared elsewhere we could lose the possibility of cancellation, but if expressions become very complex the problem of automatic symbolic simplification is not tractable in a reasonable amount of computer time. Note that saying that, "We

have a way of calculating the number," is implied if we say, "We have compiled code to produce its value in a known location in memory."

We do not consider the fact that we are, in some cases, going to want to form  $R^{-1}$  many times where the values of L, R, S, and T are fixed (so that  $R^{-1}$  is fixed) but where C and D (and hence B) change. This can be done by the two sets of steps below. The first and second sets must be done each time that L, R, S, and T change; only the second set need be done if C and D change. These steps are obtained by going back over those steps given above and writing down only those that result in "complex" expressions; that is, ones we do not wish to retain symbolically. In the steps below, all names refer to the contents of locations in memory which do not need to be stored in any particular order.

SET 1		SET 2
1.	$S+T \rightarrow R_{15}$	
2.	$-R * R_{15} \rightarrow R_{35}$	$C-R \rightarrow B_3$
3.	$1-R_{15} \rightarrow R_{45}$	
4.		$D/2 \rightarrow B_5$
5.		$-B_5 * R_{45} \rightarrow B_4$
6.		$-B_3 - R_{35} * B_5 \rightarrow B_3$
7.		$B_5 \rightarrow B_2$
8.		$1-R_{15} * B_5 \rightarrow B_1$
9.		$B_1 - B_2 \rightarrow B_1$



Thus, we use a multiplication and two additions (or subtractions) to form the new "operator inverse of A," that is a sequence of steps  $S$  which when applied to  $B$  give  $R^{-1}B$ . (i.e.  $S(B) = R^{-1}B$ ). The sequence  $S$  is given in SET 2, and takes one division, three multiplications.

Two problems have been ignored. They relate to the selection of the pivot element. In the example above it was always taken to be the next diagonal element. The choice of pivot affects both the retention of zero elements (which is obviously desirable in order to reduce the amount of work) and the numerical accuracy--in the worst case the diagonal element could be zero so that the division of the row by the diagonal element is not possible. To see the effect of bad selection of pivots on the zeros, consider the 5 x 5 matrix

1	x	x	x	x
x	1	0	0	0
x	0	1	0	0
x	0	0	1	0
x	0	0	0	1

where  $x$  represents any nonzero matrix.

If the (1,1) element is used as pivot the first time, the resulting matrix will be of the form

1	x	x	x	x
0	x	x	x	x
0	x	x	x	x
0	x	x	x	x
0	x	x	x	x

resulting in the loss of all of the initial zeros. (This is the worst case!). If, instead, the pivots had been taken in the order (2,2), (3,3), (4,4), (5,5), and (1,1), no zeros would have been lost in the process.

Numerical error requirements dictate that the pivot element may not be "too small." The exact meaning of too small is complex and beyond the scope of this report, but in most programs the largest element in the chosen column is used. (Since any row, representing an equation, can be scaled by any nonzero quantity, it is obvious that each row must be normalized in some manner to apply this criterion.) Our desire is to perform most of the inversion symbolically before the actual values are known so it is not possible to make such a choice in general.

Consequently, we must make some reasonable assumptions during the symbolic steps, and possibly note some tests that should be made during the subsequent numerical steps to verify the reasonableness of the operations. We started with either numerical values (integers or constants) or symbolic expressions. Since we cannot tell much about the value of expressions, we must assume that they could take the value zero (although it is evident that expressions such as  $X^2 + 1$  are bounded away from zero), and that we must try and use elements with numerical values as pivots first. Pivots with the value 1 or -1 are most desirable because no division is necessary. (The distinction between small integers and constants is simply the question of certainty of value. If we perform the symbolic manipulation  $(-2)*L + 2*L$  where -2 and 2 are "small integers," we know we get zero, whereas if we perform  $(-2.0)*L + 2.0*L$  where -2.0 and 2.0 are constants that have been calculated and happen to

be the same, we only know that each represents a number whose magnitude is in the range  $2.0 \pm \epsilon$ , consequently, the expression could be in the range  $\pm 2\epsilon L$ . It may prove necessary to use this fact in automatic analysis.)

The two goals of pivot selection may lead to conflicting criteria: The largest element in a column may cause the greatest destruction of zero elements. Therefore, in choosing the pivot, we will first concentrate on avoiding zero loss, then on numerical accuracy. (This may appear backwards since the end result has first to be accurate then to be rapidly obtained, but there is some flexibility in pivot choice in many problems--only a very bad choice will cause unacceptable errors--whereas bad pivot selection often increases the number of nonzeros, and hence operations, drastically.)

The choice of pivot elements has received study for sparse matrices (ones with many zeros) by a number of people. It does not appear to be possible to find the best strategy without examining all permutations of columns (which is not practical for large  $N$ ), but a number of heuristics have been tried experimentally.

Dantzig, et.al.<sup>[3]</sup> tested ten different rules on linear programming problems. Although no one rule is uniformly better, one was recommended for their class of problems. We will use it until tests have been made to determine better rules.

The rule is first to examine each column to find out how many nonzero elements it contains. The subset  $\{C\}$  of those columns with the minimum number of nonzero elements is chosen. Of all rows with nonzero elements in the columns  $\{C\}$ , the first row  $A$  with the minimum number of nonzero elements is chosen. The pivot element is the first common nonzero element of  $A$  and  $\{C\}$ .

We will modify the last sentence to "the pivot element is the most simple element common to A and {C}, where most simple means an order in which 1 and -1 are simplest, followed by small integers, variables, constants reasonably different from zero, and lastly symbolic expressions." Note that after the first pivot has been chosen and the necessary row operations performed, the next choice should be made on the basis of the of reduced matrix, ignoring the rows already chosen as pivot rows.

We also wish to bias the selection in favor of rows with zeros in the B matrix. Consequently, in choosing the row when there are several rows with the same number of elements, we favor the first with a zero row in B.

We will illustrate the mechanization of this technique on our earlier example. To simplify the operations of searching for the best rows and columns we will initially count the number of nonzero elements in rows and columns:

NON ZERO COUNT	3	4	1	1	3	1
3	1	-1	0	0	x	1
2	0	L	0	0	-L	0
3	R	-R	1	0	0	C
2	1	0	0	1	0	1
2	0	1	0	0	1	D

x stands for an arbitrary number.

The first step gives columns 3 and 4 with one element, Row 4 is chosen, so the first pivot is the (4,4) element. No operations are required on the matrix, but not the element in the (4,1) position is no longer considered so the form is

STEP NUMBER		SC			1			
NONZERO COUNT ZC								
		2	4	1	1	3		
SR	ZR							
--	3	1	-1	0	0	x	1	
--	2	0	L	0	0	-L	0	
--	3	R	-R	1	0	0	C	
1	2	1	0	0	1	0	1	
--	2	0	1	0	0	1	D	

The order in which rows and columns are chosen is also specified in vectors SR and SC. The next pivot is (3,3) which also requires no row operations, but ZC(1) is reduced to 1, ZC(2) to 3, while SR(3) = SC(3) = 2. Consequently, column 1 is the next choice, and there is only one nonzero element in that column in the rows not yet handled, so it is the pivot; that is, the (1,1) element. Now the matrix is

		SC			3	-	2	1	-
		ZC			1	2	1	1	
SR	ZR								
3	3		1	-1	0		0	x	1
-	2		0	L	0		0	-L	0
2	3		R	-R	1		0	0	C
1	2		1	0	0		1	0	1
-	2		0	1	0		0	1	D

Either of the two remaining columns is acceptable and the row ZR is the same, but row 2 has a zero value for B. Therefore, we choose either (2,2) or (2,5). In this case it is immaterial since both are

variables. If (2,5) were chosen the result would be to set  $SR(2) = 4$ ,  $SC(5) = 4$ ,  $R(2,2) = -1$ ,  $R(2,5) = 1$ ,  $R(5,2) = 2$ , and  $R(5,5) = 0$ . The final pivot is  $R(5,2)$  meaning that its value of 2 must be used to generate a divide by 2 on the fifth element of B. Since this has not been stored yet, we generate  $D/2 \rightarrow B_5$  as one of the second sets of steps. Note that apart from the formation of  $S+T \rightarrow R_{15}$ , no first step processing must be done when the values of L, R, S, and T are changed. (This is a fortunate example in this respect because R is almost triangular after permutation.)

The back substitution step requires that we handle the pivots in the reverse order. Thus, we find the fifth pivot in

		3	5	2	1	4	
		1	2	1	1	2	
3	3	1	-1	0	0	x	1
4	2	0	-1	0	0	1	0
2	3	R	-R	1	0	0	C
1	2	1	0	0	1	0	1
5	2	0	1	0	0	0	$B_5$

It is the (5,2) element. All elements in the second column must be eliminated by generating the steps

$$1 + B_5 \rightarrow B_1$$

$$C + R*B_5 \rightarrow B_3$$

and modifying the sixth column of R to read

$$[B_1 \ B_5 \ B_3 \ 1 \ B_5]^T$$



The next pivot element is (2,5) and the process generates

$$B_1 - x*B_5 \rightarrow B_1$$

for the one nonzero nonpivot element in the fifth column.

Continuing, we generate

$$B_3 - R*B_1 \rightarrow B_3$$

$$1 - B_1 \rightarrow B_4$$

$$(\text{and set } B(4) = R(4,6) = B_4)$$

to complete the process. The reorganization saved two steps in SET 1 and one step in SET 2. For larger matrices it could be much more significant. Note that the solution obtained needs reordering. B(1) to B(5) corresponds to x(1), x(5), x(3), x(4), x(2) because not all pivots were diagonal elements.

There is some choice of storage organization. For each nonzero element it is desirable to store its type (complexity), value (or pointer to memory location containing a character string or address of place in which it will be available at execution time), and its row and column number. Since it is necessary to scan both rows and columns, a chain in both row and column directions is desirable. For each row and column we need the nonzero counts, we need to know if a row has already been handled and we need to be able to trace backwards through the pivot elements. The latter could be handled by a chain through these elements, but that would make them unequal in size from the nonpivot elements, complicating storage assignment. Consequently, it appears better to construct



backward chains through the row and column vector elements which store the number of nonzeros. (These are not used after the row and column take part in a pivot.) Thus, the original form of R is a compacted matrix represented by row and column vectors containing the chain pointers. The result of "inverting" R will be another compacted matrix whose elements may be numeric, symbolic, or addresses of locations containing the result. (In the latter case code will have been compiled to produce the result.) The "lower" part of the matrix can represent the elimination steps yet to be performed (those that were not either produced as code or performed numerically), while the diagonal represents the division steps and the "upper" triangular part the back substitution steps.

Singularities will be detected in R when we find that the minimum number of nonzero elements in a remaining column is zero. That column should be ignored until the elimination process has been completed. At that time there will be a number of zero rows of R left, and a number of columns not used in the pivoting process. Those rows containing nonzero elements on the righthand side of the reduced B can be eliminated. The equations corresponding to the others can be placed in the nonlinear set (3.2) and associated with variables in columns that were not used as pivots.

Considerable theoretical study remains to be done to determine if this procedure will always work if the network is reasonably well specified and to find methods that will determine when the network is improperly specified.

## 5. CONCLUSIONS

A system which is currently being implemented has been outlined. When complete, this system will accept arbitrary definitions of elements, and be capable of doing transient analyses of correctly specified networks. Most incorrect networks will be detected. There are many problems and extensions to be analyzed, and it is planned to continue the analysis throughout the implementation phase. The major problem is that of relating detected errors to the source of the error in user notation. Singularities in the system of equations could be caused by any one of many elements or connections, and it is necessary to tell the user the most likely causes.

Extensions that need to be analyzed include the following. Discontinuous functions possibly corresponding to the use of discrete variables mixed and logical simulation problems, would allow a useful class of elements to be introduced. Probabalistic functions would also help in many simulations, although they would greatly complicate the numerical algorithms. Time delay elements are important in control systems and should be allowed, although they also complicate good numerical integration algorithms. In more generality, convolution functions would allow integral equations to be tackled. Finally, arbitrary arithmetic functions defined by subroutines would be desirable; these raise many problems because the derivatives of the functions are required in the analysis presently used.

REFERENCES

1. Baskin, H. B., Morse, S. P. "A Multi-Level Modeling Structure for a Graphic Design Facility," IBM Research Report RC 1990.
2. Brennan and Silberberg "Two Continuous System Modeling Programs," IBM Systems Journal, 6, #4, (1967).
3. Dantzig, G. B., et.al. "Sparse Matrix Techniques in Two Mathematical Programming Codes," Tech. Report #69-1, (Jan. 1969). Operations Research House, Stanford University.
4. Dill, C., Ellis, C., Gear, C., Ratliff, K. "ODESSY--Ordinary Differential Equation Solver System," Department of Computer Science File No. 779, University of Illinois, Urbana, Illinois.
5. Gear, C. W. "The Automatic Integration of Stiff Ordinary Differential Equations," Proceedings IFIPS '68, pp. A81-A85.



U. S. ATOMIC ENERGY COMMISSION  
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR  
DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

( See Instructions on Reverse Side )

AEC REPORT NO. COO-1469-0115	2. TITLE AN INTERACTIVE GRAPHIC MODELING SYSTEM
---------------------------------	--

3 TYPE OF DOCUMENT (Check one):

- ☒ a. Scientific and technical report
- ☐ b. Conference paper not to be published in a journal:  
Title of conference \_\_\_\_\_  
Date of conference \_\_\_\_\_  
Exact location of conference \_\_\_\_\_  
Sponsoring organization \_\_\_\_\_
- ☐ c. Other (Specify) \_\_\_\_\_

4 RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- ☒ a. AEC's normal announcement and distribution procedures may be followed.
- ☐ b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
- ☐ c. Make no announcement or distribution.

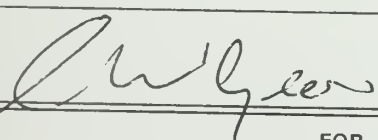
5 REASON FOR RECOMMENDED RESTRICTIONS:

6 SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear, Professor  
of Computer Science and Applied Mathematics

Organization Department of Computer Science  
University of Illinois  
Urbana, Illinois 61801

Signature



Date

April 7, 1969

FOR AEC USE ONLY

7 AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

8 PATENT CLEARANCE:

- ☐ a. AEC patent clearance has been granted by responsible AEC patent group.
- ☐ b. Report has been sent to responsible AEC patent group for clearance.
- ☐ c. Patent clearance not required.















1973 23 JAN



UNIVERSITY OF ILLINOIS-URBANA  
510.84 IL6R no. C002 no.316-322(1968  
Internal report /



3 0112 088398539